**The following steps don’t include information about documenting public interface with argparse since documenting using argparse is the next subtask for this assignment (i.e. self.\_parser and some other variables are not present here).**

**Step 1: Get an informal list of the responsibilities of your objects.**

* Store country’s name, area, and population after an instance is created.
* Find the country with the largest area among the instances that were defined, use list classes.
* Find the country with the largest population among the instances that were defined, use list classes.
* Find the country with the largest population density among the instances that were defined, use list classes.
* Find the country with the largest area among the instances that were defined, use dictionary classes.
* Find the country with the largest population among the instances that were defined, use dictionary classes.
* Find the country with the largest population density among the instances that were defined, use dictionary classes.

**Step 2: Specify the public interface.**

From the task description we observe that we need to include several Class variables to implement most of the methods

class Country:

\_listNames = []

\_listPopulations = []

\_listAreas = []

\_countriesDict = {}

def \_\_init\_\_(self, name, population, area):

def getCountryLargestAreaListImplement(self):

def getCountryLargestPopulationListImplement(self):

def getCountryLargestPopDensityListImplement(self):

def getCountryLargestAreaDictImplement(self):

def getCountryLargestPopulationDictImplement(self):

def getCountryLargestPopDensityDictImplement(self):

**Step 3: Document the public interface.**

class Country:

# a list to keep track of names of the countries that were instantiated

\_listNames = []

# a list to keep track of populations of the countries that were instantiated

\_listPopulations = []

# a list to keep track of areas of the countries that were instantiated

\_listAreas = []

# a dictionary with keys being the names of countries that were instantiated and

# values being tuples with population and area of a country at the first

# and second positions respectively

\_countriesDict = {}

## Constructs an object for a country

# @param name: name of a country

# @param population: population of a country

# @param area: area of a country

#

def \_\_init\_\_(self, name, population, area):

## Finds the country with the largest area among the instances that were defined,

# uses Class variable of type list for this purpose.

# @return name of the country with the largest area among the instances that were # defined

def getCountryLargestAreaListImplement(self):

## Finds the country with the largest population among the instances that were

# defined, uses Class variable of type list for this purpose.

# @return name of the country with the largest population among the instances that

# were defined

def getCountryLargestPopulationListImplement(self):

## Finds the country with the largest population density among the instances that

# were defined, uses Class variable of type list for this purpose.

# @return name of the country with the largest population density among the

# instances that were defined

def getCountryLargestPopDensityListImplement(self):

## Finds the country with the largest area among the instances that were defined,

# uses Class variable of type dictionary for this purpose.

# @return name of the country with the largest area among the instances that were # defined

def getCountryLargestAreaDictImplement(self):

## Finds the country with the largest population among the instances that were

# defined, uses Class variable of type dictionary for this purpose.

# @return name of the country with the largest population among the instances that # were defined

def getCountryLargestPopulationDictImplement(self):

## Finds the country with the largest population density among the instances that

# were defined, uses Class variable of type dictionary for this purpose.

# @return name of the country with the largest population density among the

# instances that were defined

def getCountryLargestPopDensityDictImplement(self):

**Step 4: Determine instance variables.**

We need to store name, area, and population for each object. Thus, we have three instance variables:

\_name

\_population

\_area

**Step 5: Implement the constructor.**

def \_\_init\_\_(self, name, population, area):

self.\_name = name

self.\_population = population

self.\_area = area

# list implementation

Country.\_listNames.append(name)

Country.\_listPopulations.append(population)

Country.\_listAreas.append(area)

# dict implementation

# key: country name, value: tuple with population at the first position,

# and area at the second position

#

Country.\_countriesDict[name] = (population, area)

**Step 6: Implement the methods.**

def getCountryLargestAreaListImplement(self):

# gets a position in a list of areas of country with

# the largest area

pos = Country.\_listAreas.index(max(Country.\_listAreas))

# uses the index to extract the name of the country

return Country.\_listNames[pos]

def getCountryLargestPopulationListImplement(self):

# gets a position in a list of areas of country with

# the largest population

pos = Country.\_listPopulations.index(max(Country.\_listPopulations))

# uses the index to extract the name of the country

return Country.\_listNames[pos]

def getCountryLargestPopDensityListImplement(self):

# create local variables to find an index of a contry with

# the largest density

# the default value is equal to 0

# to avoid a possibility of index being out of range

pos = 0

# the defulat value is equal to -1

# so that in any case it is going to be replaced with

# an actual value for density

max\_density = -1

# we extract the number of countries in order to iterate over

# each instance

no\_countries = len(Country.\_listNames)

for i in range(no\_countries):

# calculate density for each country

density = Country.\_listPopulations[i] / Country.\_listAreas[i]

if density > max\_density:

pos = i

max\_density = density

return Country.\_listNames[pos]

def getCountryLargestAreaDictImplement(self):

name = None

# max\_area being equal to -1 initially

# so that it will be changed after the first iteration of the for loop

max\_area = -1

for country\_name in Country.\_countriesDict:

if Country.\_countriesDict[country\_name][1] > max\_area:

name = country\_name

max\_area = Country.\_countriesDict[country\_name][1]

return name

def getCountryLargestPopulationDictImplement(self):

name = None

max\_population = -1

for country\_name in Country.\_countriesDict:

if Country.\_countriesDict[country\_name][0] > max\_population:

name = country\_name

max\_population = Country.\_countriesDict[country\_name][0]

return name

def getCountryLargestPopDensityDictImplement(self):

name = None

max\_pop\_density = -1

for country\_name in Country.\_countriesDict:

pop\_density = Country.\_countriesDict[country\_name][0] / Country.\_countriesDict[country\_name][1]

if pop\_density > max\_pop\_density:

name = country\_name

max\_pop\_density = pop\_density

return name

**Step 7: Test your class.**

Norway = Country("Norway", 5.4, 385207)

Finland = Country("Finland", 5.5, 338462)

assert Finland.getCountryLargestAreaListImplement() == "Norway"

assert Norway.getCountryLargestAreaDictImplement() == "Norway"

assert Finland.getCountryLargestPopDensityListImplement() == "Finland"

assert Finland.getCountryLargestPopDensityDictImplement() == "Finland"

Sweden = Country("Sweden", 10.4, 528447)

assert Norway.getCountryLargestPopulationDictImplement() == "Sweden"

Denmark = Country("Denmark", 5.9, 42920)

assert Sweden.getCountryLargestAreaListImplement() == "Sweden"

assert Denmark.getCountryLargestPopulationListImplement() == "Sweden"

assert Finland.getCountryLargestPopDensityListImplement() == "Denmark"